**Facial Attribute Analysis Using Deepface and CNN**

A PROJECT REPORT

Submitted by

## Venkata Harsha Vardhan Beeraka

**19MIS1203**

in partial fulfillment for the award of the degree of

Master of Technology

in

Software Engineering (5 Year Integrated Programme)

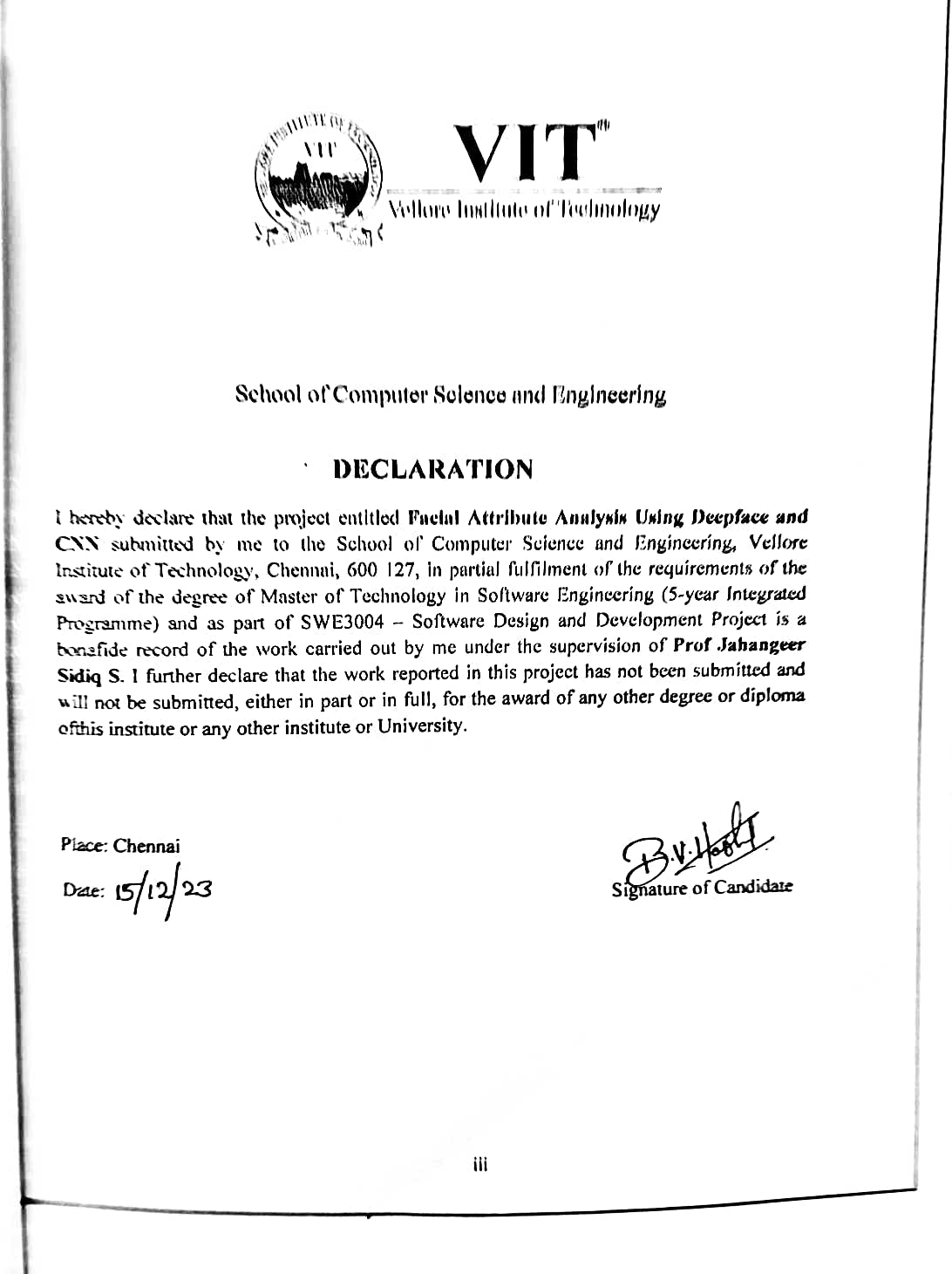




## School of Computer Science and Engineering

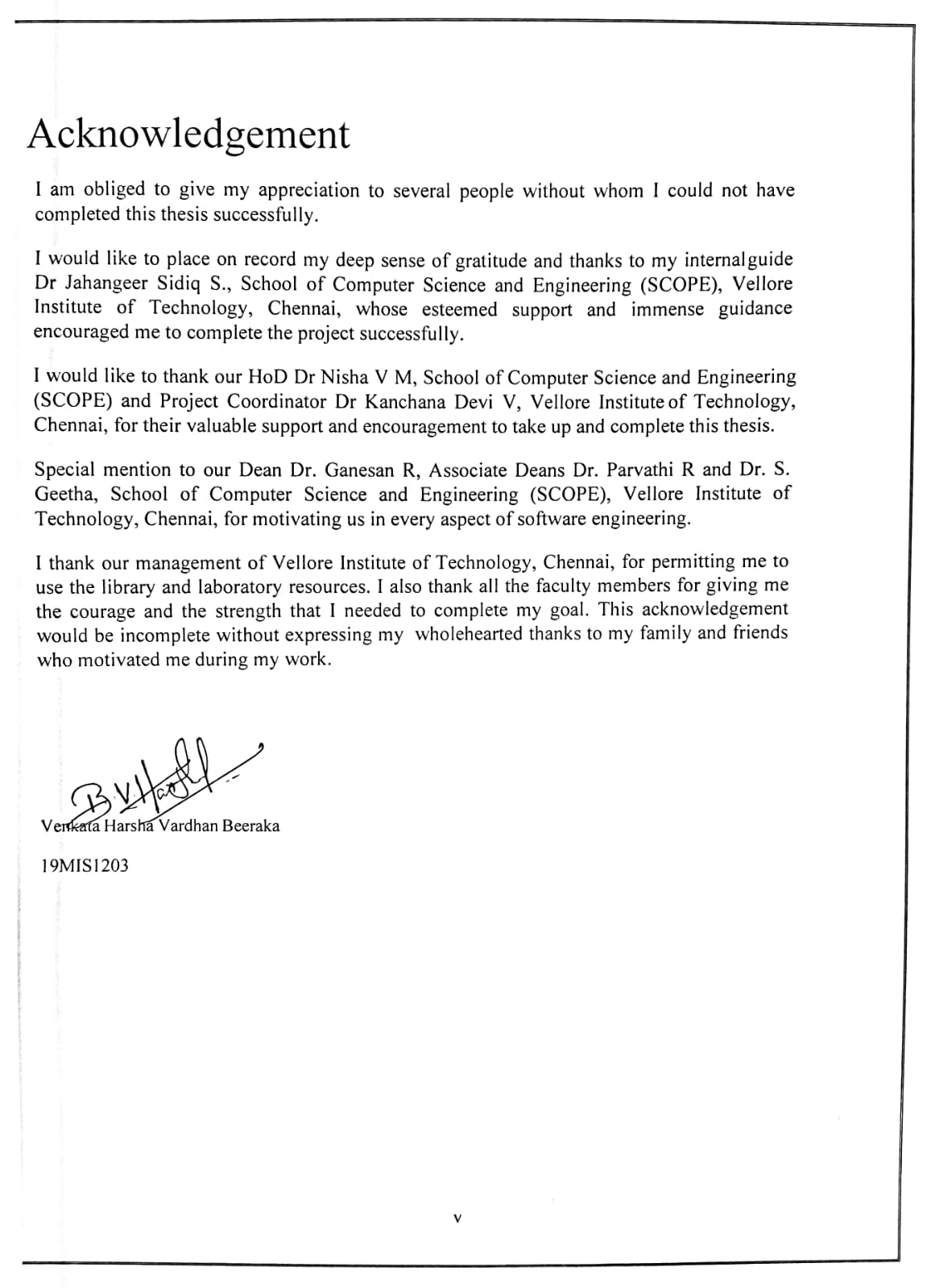
Vellore Institute of Technology

Vandalur - Kelambakkam Road, Chennai - 600 127 November - 2023



A close-up of a certificate

Description automatically generated



# Abstract

# This project is a detailed comparison between two facial attribute analysis models: the well-known DeepFace model and a custom Convolutional Neural Network (CNN) algorithm. The primary focus of this analysis is on their capabilities to accurately predict three key attributes: age, gender, and race. The effectiveness of these models is evaluated using specific criteria: Mean Absolute Error (MAE) for age prediction, and accuracy percentages for gender and race prediction.

# The results show that the custom model achieved a MAE of 14.67 for age prediction, indicating the average error in years. In terms of gender and race prediction, this model reached accuracy levels of 68.50% and 44.67%, respectively. On the other hand, the DeepFace model displayed robust performance with a lower age prediction MAE of 12.32, suggesting a closer approximation to the actual ages. It scored higher accuracy in predicting gender (77.38%) and race (61.60%).

# The findings from this comparison suggest that the DeepFace model outperforms the custom CNN in all three tested attributes. This superior performance of DeepFace highlights its advanced algorithms and sophisticated data processing capabilities in facial attribute analysis. The study contributes valuable insights into the field of facial recognition technology, especially in the accurate prediction of age, gender, and race.

# **CONTENTS**

**CHAPTERS PAGE NO**

Declaration……………………………………………………………………………….. iii

Certificate………………………………………………………………………………….iv

Acknowledgement………………………………………………………………………. v

Abstract……………………………………………………………………………………vi

1. [Introduction](#_bookmark1)
   1. Background……………………………………………………………………..... 10
   2. Problem Statement……………………………………………………………….. 11
   3. Motivation……………………………………………………………………….. 11
   4. Challenges…………………………………………………………………….... 12
2. Planning & Requirements Specification
   1. System Planning……………………………………………………………….. 13
   2. Requirements………………………………………………………………….. 14
      1. User Requirements……………………………………………………… 14
      2. Non-Functional Requirements…………………………………………. 14
   3. Requirements…………………………………………………………………. 15
      1. Hardware Requirements………………………………………………. 15
      2. Software Requirements………………………………………………... 15
3. System Design
   1. Module Explanation (Methodology)…………………………………………….. 16

3.2 System Architecture……………………………………………………………… 19

1. Implementation Of System
   1. General………………………………………………………………………………20
   2. Code Implementation………………………………………………………………..21
2. Results And Discussions………………………………………………………….. 30
3. Conclusion And Future Work……………………………………………………...42
4. References………………………………………………………………………… 43

## CHAPTER -1

**INTRODUCTION**

Facial Attribute Analysis is a pivotal aspect of computer vision in artificial intelligence, identifies and quantifies various human facial attributes such as age, gender, emotion, and ethnicity. Its applications span from security through facial recognition to customer experience enhancement in retail.

A prime example of this technology is Facebook AI's 'Deepface,' which achieves near-human accuracy in facial recognition. Deepface creates a 3D face model, and then normalizes it to address pose and illumination variations and uses a deep neural network for precise feature extraction and classification. Facial attribute analysis is a crucial component in the field of artificial intelligence, particularly in computer vision. It plays a vital role in identifying and quantifying human facial features such as age, gender, emotion, and ethnicity. This technology has significant applications in various sectors, including security, marketing, and personal identification. Since the dawn of Deep Learning, especially Convolutional Neural Networks (CNNs), has significantly advanced this field. CNNs excel in image recognition due to their ability to learn feature representations from raw data, streamlining the facial attribute analysis process.

Facebook AI's 'Deepface' stands out as prime example of advanced facial attribute analysis. Deepface utilizes a 3D face modeling technique to normalize facial images for pose and lighting variations and employs a deep neural network for efficient and precise feature extraction and classification.

This technology allows for real-time processing and analysis of facial attributes. Using CNN models trained on extensive datasets, Deepface can quickly and accurately identify facial features from images or live feeds. The methodology involves training CNN models on diverse facial datasets. The architecture of Deepface and similar systems typically comprises several layers of convolution, pooling, and fully connected layers.

##### 1.1. Background:

Shi et al. summarized modern face-based age estimation methods and proposed an attention-based convolution (ABC) age estimation framework, notably improving the accuracy of face age estimation. This framework represents a novel approach in the field.

A study by Yang et al. in 2023 focused on the security concerns in face verification and recognition, especially when retrieving and identifying faces from large-scale databases or live surveillance videos. The study emphasizes the importance of security in these widely used techniques. Li and Deng (2022) conducted a comprehensive survey on deep FER, including datasets and algorithms. They reviewed various deep neural networks and training strategies designed for FER, addressing challenges like over-fitting and expression-unrelated variations. Additionally, Kong et al. (2022) proposed a real-time facial expression recognition method based on iterative transfer learning and an efficient attention network (EAN), aimed at edge resource-constrained scenarios. Zhao et al. proposed a style Attention based Global-local Aware GAN for Personalized Facial Caricature Generation, integrating a landmark-based warp controller for personalized shape exaggeration and a style-attention module to enhance the quality of generated caricatures. Monteiro et al. provided evidence that facial asymmetry measurements might not be fully comparable across different populations. They analyzed differences between the Chicago Face Database and the LACOP Database, finding consistent disparities. Their study highlights the need for multi-ethnic face image databases for specific population research, important for both face perception research and computer vision.

##### 1.2. Problem Statement:

These technologies have notably advanced AI's ability in facial recognition, offering enhanced accuracy and efficiency. Deepface solidifies the application of CNNs in achieving advanced facial recognition, pushing the boundaries of AI capabilities. The rise of these technologies also brings ethical challenges, particularly in privacy and responsible use. Ongoing research in this area not only drives technological innovation but also underscores the importance of ethical considerations in AI development.

##### 1.3. Motivation:

The focus on using Deepface and CNNs for facial attribute analysis is driven by their potential to change how AI approaches facial recognition. These technologies don't just offer better accuracy, but also open many new uses, from improving security to more personalized interactions.

Exploring these advanced tools aims to make the most of their tech potential while also carefully considering their ethical implications. As we dive deeper, we're focused on using these advancements responsibly, reshaping how facial recognition works and staying aware of the ethical and social impacts of such powerful AI. This exploration aims to both understand and use these advancements carefully, keeping in mind the ethical aspects of such powerful AI tools. As we move forward, the goal is to use these technologies responsibly, not only advancing AI's ability to recognize and interpret faces but also considering the broader impact on society and individual privacy.

##### 1.4. Challenges:

##### Age and gender are not static categories, and there is considerable variability within each group. Furthermore, the race of individuals only increases the complexity. People of the same gender or age can exhibit diverse physical characteristics, making accurate detection challenging. Cultural and ethnic diversity can significantly impact the accuracy of age and gender detection algorithms. Facial features, clothing styles, and grooming practices vary across different cultures, which can lead to biased or inaccurate predictions. Some facial features may not exclusively indicate gender or age. For instance, hairstyles, makeup, and clothing choices can be confusing for the algorithm, leading to misclassifications. Age-related changes in appearance are not always linear or consistent. Many lifestyle factors, health, and genetics can influence the ageing process, making it challenging to accurately estimate a person's age based on appearance alone. Datasets used for training gender and age detection models may suffer from imbalances, with insufficient representation of certain age groups, genders, or ethnicities. This can lead to biased models that perform well on dominant groups but poorly on underrepresented ones.

## CHAPTER -2

**PLANNING & REQUIREMENTS SPECIFICATION**

##### 2.1. System Planning:

SEP01\_literature survey

SEP24 \_Model planning

NOV02\_ Partial implementation

NOV30\_ Complete implementation

**Back End Module Diagram:**

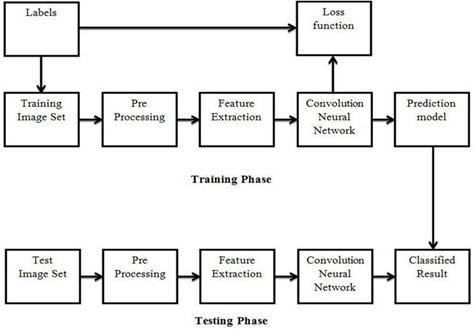


Fig. 1. Back-end Model Diagram for Custom CNN Models

##### 2.2. Requirements:

**2.2.1. User Requirements:**

*Functional:* Accurate identification and analysis of facial attributes, with consistency across diverse data.

*Usability:* An intuitive interface that simplifies image uploading and result viewing for all user levels.

*Performance:* High processing speed and accuracy, capable of handling large data volumes efficiently.

*Compatibility:* Seamless integration with various data formats and existing AI tools.

##### 2.2.2. Non-Functional Requirements:

*Scalability:* Ability to efficiently handle increasing data volumes and user requests.

*Reliability:* Consistent and accurate operation with minimal downtime.

*Efficiency:* Optimized use of computational resources for swift and precise processing.

*Maintainability:* Easy updates and enhancements with minimal system disruption.

*Portability:* Adaptability to different platforms and operational environments.

*Security:* Strong measures to safeguard sensitive data against unauthorized access.

*Compliance:* Adherence to legal and ethical standards in data protection and AI use

*User Experience:* Responsive interface and clear outputs for a positive user experience.

##### 2.3. System Requirements:

##### 2.3.1. Hardware Requirements:

*Processing Power:* A minimum of an Intel Core i5 processor or equivalent, with a preference for i7 or higher for more intensive tasks.

*RAM:* At least 8GB of RAM, though 16GB or more is recommended for handling large datasets and complex models efficiently.

*Graphics Card:* A dedicated GPU, such as NVIDIA GTX 1060 or better, is crucial for accelerating deep learning processes.

*Storage:* A minimum of 256GB SSD storage for faster data access and processing. Additional HDD storage is beneficial for large datasets.

*Software:* The latest versions of Python, TensorFlow, or PyTorch, and other relevant deep learning libraries.

*Operating System:* A stable and updated version of either Windows 10 or Higher

Adhering to these specifications ensures that users can effectively run and experiment with Deepface and CNN-based systems for Facial Attribute Analysis, without encountering performance bottlenecks.

##### 2.3.2. Software Requirements:

Operating System : Windows 10/11 Server side Script : HTML, CSS Programming Language : Python

Libraries : Pandas, Deepface, OS, matplotlib, NumPy, ScikitLearn

IDE/Workbench : Visual Studio Code, Flask Framework

Web Browser : Compatibility with either Google Chrome, Microsoft Edge

Technology : Python 3.11.5+

## CHAPTER – 3

**SYSTEM DESIGN**

**3.1 Module Explanations (Methodology)**

Data Collection Module

Data Preprocessing Module

Convolutional Neural Network (CNN) Architecture Module

Flask Integration Module

User Interface Module

**3.1.1. Data Collection Module:**

*Objective:* Gather a wide range of facial images, covering diverse attributes such as age, gender, ethnicity, and emotions.

*Methodology:*

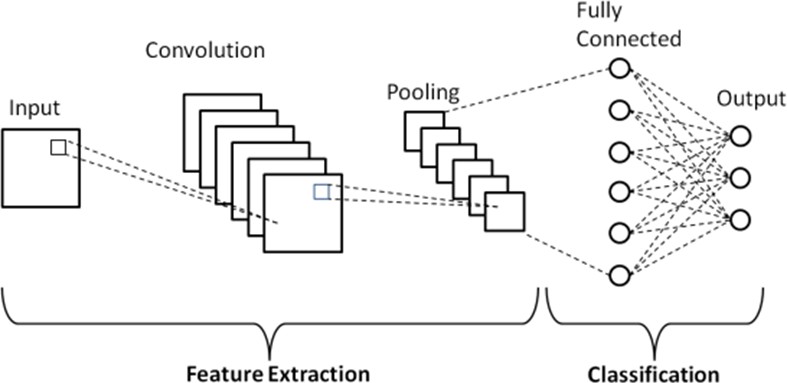
Source facial image dataset from UTKFaces. We must make sure that the data is meticulously labeled with detailed attributes to facilitate precise model training. In the realm of facial attribute analysis, the data analyst plays a crucial role. Key responsibilities include selecting appropriate datasets, extracting meaningful information, analyzing facial features, and applying statistical methods to understand different facial attributes. This process is integral to developing a robust CNN model that can accurately identify and analyze various facial characteristics.

**3.1.2. Data Pre-processing Module:**

*Objective:*Prepare the collected images for model training by applying necessary preprocessing techniques.

*Methodology:*

Now, we use image processing libraries such as OpenCV and scikit-image for tasks like resizing images to a standardized format (e.g., 224x224), normalization of pixel values to a specific range, and then, we make the pre-processed images go through the CNN Module.

[](https://www.mdpi.com/2076-3417/13/1/131?type=check_update&version=1)**3.1.3. Convolutional Neural Network (CNN) Architecture Module:**

### Methodology:

Fig. 2. Architecture of CNN Model

**CNN Architecture Design:**

Now, we use TensorFlow and Keras to create a deep learning model, that is mainly concerned with recognizing the age, gender, race of a person. And later, we configure the model architecture with convolutional layers for feature extraction, pooling layers for down-sampling, and dense layers for classification.

### **Training Phase:**

We utilize a portion of the collected dataset for training the CNN model. Here, we separate divide the training set to be 45% of the total size of the dataset. We must ensure that people from all the different age groups and races must be included. Define appropriate loss functions, such as binary cross entropy and categorical cross entropy, also, we select suitable optimizers, (here) Adam, for model training. Train the model over 10 epochs, adjusting hyperparameters based on validation performance. Implement data augmentation techniques during training to enhance model generalization.

### **Validation Phase:**

Assess the model's performance on a separate validation set to ensure it generalizes well to unseen data. Allocate another 20% of the entire dataset for validation. Monitor metrics such as accuracy, recall during validation to fine-tune the model.

### **Testing Phase:**

Finally, we evaluate the trained CNN model on a dedicated test dataset (the remaining 35%) not used during training or validation. Measure key metrics, including accuracy, MAE to gauge the model's real-world performance. Analyze the model's ability to correctly classify the person’s age, gender, and race.

### **3.1.4. Flask Integration Module:**

*Objective:* Build a user-friendly web interface for real-time interaction with the Facial Attribute Analysis system.

### *Methodology:*

Use Flask, a lightweight Python web framework, to create routes and endpoints for handling user requests. Integrate the trained CNN model into the Flask application to make predictions on user-uploaded images. Implement error handling and user feedback mechanisms.

### **3.1.5. User Interface Module:**

*Objective:*Develop an intuitive and responsive user interface for seamless interaction.

### *Methodology:*

Implement HTML for the structure, CSS for styling, and JavaScript for dynamic behavior. Design a user- friendly interface with features like file upload, result display, and user feedback. Ensure responsiveness for different devices and browsers.

* 1. **ARCHITECTURE DIAGRAM:**

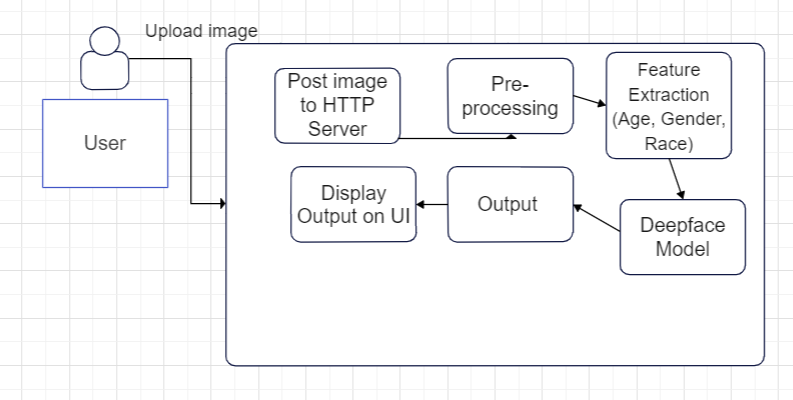
****

Fig. 3. System Design

## CHAPTER – 4

**IMPLEMENTATION OF SYSTEM**

**4.1 GENERAL**

The implementation of the Facial Attribute Analysis System involves a synergistic integration of Python Flask for the backend, leveraging its web application framework capabilities, and a suite of powerful Python libraries for deep learning-based model development and image processing techniques.

**Python Flask Framework:**

**Backend Development:** The backend of the system is developed using the Python Flask framework, a lightweight and versatile web application framework. Flask provides a simple yet extensible architecture, making it an excellent choice for developing web applications. It facilitates the creation of routes, handling HTTP requests, and managing interactions between the user interface and the underlying logic.

**Web Application Structure:** The Flask application is structured to handle different functionalities, such as user image uploads, real- time prediction requests, and result. It is designed to provide a seamless and intuitive user experience through a web interface.

**Deep Learning Libraries:** For the implementation of the Facial Analysis model, the deepface module is employed. The foundations of the deepface are *TensorFlow and Keras.* TensorFlow is an open-source machine learning library, and Keras serves as a high-level neural networks API running on top of TensorFlow. This combination allows for the construction and training of complex Convolutional Neural Networks (CNNs) for image classification tasks.

**Real-time Interaction:**

The Flask application provides a user-friendly interface accessible through web browsers. HTML, CSS, are used to design visually appealing front-end components. Users can upload their pictures from the system or take a picture trough the webcam and visualize the results.

**Integration with Model:**

The Flask application integrates with the deepface model. When a user uploads an image, the model processes the input and provides instant predictions, contributing to the real- time nature of the system.

**4.2 Code Implementation:**

**Model Code:**

**cnn.py:**

# pre-processing functions

def preprocess\_image(image\_path, size=(224, 224)):

    img = cv2.imread(image\_path)

    img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

    img = cv2.resize(img, size)

    return img\_to\_array(img) / 255.0

def parse\_labels(filename):

    parts = os.path.basename(filename).split("\_")

    if len(parts) < 3:

        return None

    return int(parts[0]), int(parts[1]), int(parts[2])

def data\_generator(file\_paths, batch\_size, preprocess\_fn, label\_fn):

    while True:

        batch\_paths = np.random.choice(a=file\_paths, size=batch\_size)

        batch\_input = []

        batch\_output = []

        for input\_path in batch\_paths:

            labels = label\_fn(input\_path)

            if labels is None:

                continue

            image = preprocess\_fn(input\_path)

            batch\_input.append(image)

            batch\_output.append(labels)

        yield np.array(batch\_input, dtype='float32'), np.array(batch\_output)

race\_mapping = {

    0: "White",

    1: "Black",

    2: "Asian",

    3: "Indian",

    4: "Others"

}

# Data Preparation

images\_directory = 'E:/capJC/paper/part1'

img\_paths = glob(os.path.join(images\_directory, "\*.jpg"))

# Splitting dataset into training, validation, and testing

train\_paths, test\_paths = train\_test\_split(img\_paths, test\_size=0.45, random\_state=42)

valid\_paths, test\_paths = train\_test\_split(test\_paths, test\_size=(20/45), random\_state=42)

# Define batch size

batch\_size = 128  # i.e. no. of images that is gonna be processed, at one time

import os

def extract\_labels\_from\_filename(filename, trashy\_file='trashy.csv'):

    try:

        parts = os.path.basename(filename).split('\_')

        if len(parts) < 4 or not parts[0].isdigit() or not parts[1] in ['0', '1'] or not parts[2].isdigit():

            raise ValueError("Invalid filename format")

        age = int(parts[0])

        gender = int(parts[1])  # 0 for male; 1 for female

        race = int(parts[2])  # Assuming integer labels for races

        return age, gender, race

    except ValueError as e:

        with open(trashy\_file, 'a') as file:

            file.write(filename + '\n')

        print(f"Error processing file {filename}: {e}")

        return None, None, None

# Age Model

age\_model = Sequential([

    Conv2D(32, (3, 3), activation='relu', input\_shape=(224, 224, 3)),

    MaxPooling2D(2, 2),

    Flatten(),

    Dense(128, activation='relu'),

    Dense(1, name='age\_output')

])

age\_model.compile(optimizer='adam', loss='mse', metrics=['mae'])

# Gender Model

gender\_model = Sequential([

    Conv2D(32, (3, 3), activation='relu', input\_shape=(224, 224, 3)),

    MaxPooling2D(2, 2),

    Flatten(),

    Dense(128, activation='relu'),

    Dense(1, activation='sigmoid', name='gender\_output')

])

gender\_model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

# Race Model

race\_model = Sequential([

    Conv2D(32, (3, 3), activation='relu', input\_shape=(224, 224, 3)),

    MaxPooling2D(2, 2),

    Flatten(),

    Dense(128, activation='relu'),

    Dense(len(race\_mapping), activation='softmax', name='race\_output')

])

race\_model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

def train\_age\_generator(train\_paths, batch\_size):

    while True:

        batch\_paths = np.random.choice(a=train\_paths, size=batch\_size)

        batch\_images = []

        batch\_ages = []

        for file\_path in batch\_paths:

            age, \_, \_ = extract\_labels\_from\_filename(os.path.basename(file\_path))

            if age is None:

                continue  # Skip this file if age is None

            image = preprocess\_image(file\_path)

            batch\_images.append(image)

            batch\_ages.append(age)

        yield np.array(batch\_images), np.array(batch\_ages)

def valid\_age\_generator(valid\_paths, batch\_size):

    while True:

        batch\_paths = np.random.choice(a=valid\_paths, size=batch\_size)

        batch\_images = []

        batch\_ages = []

        for file\_path in batch\_paths:

            age, \_, \_ = extract\_labels\_from\_filename(os.path.basename(file\_path))

            if age is None:

                continue  # Skip this file if age is None

            image = preprocess\_image(file\_path)

            batch\_images.append(image)

            batch\_ages.append(age)

        yield np.array(batch\_images), np.array(batch\_ages) #Similar code for gender and race

**The application:**

**app.py:**

from flask import Flask, render\_template, request

from werkzeug.utils import secure\_filename

from deepface import DeepFace

import base64

import os

import uuid

import traceback

app = Flask(\_\_name\_\_)

app.config['UPLOAD\_FOLDER'] = 'static/uploads'

app.config['MAX\_CONTENT\_LENGTH'] = 16 \* 1024 \* 1024  # Limit file size to 16MB

os.makedirs(app.config['UPLOAD\_FOLDER'], exist\_ok=True)

ALLOWED\_EXTENSIONS = {'png', 'jpg', 'jpeg', 'gif'}

def allowed\_file(filename):

    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED\_EXTENSIONS

@app.route('/')

def home():

    return render\_template('home.html')

@app.route('/upload', methods=['POST'])

def upload\_file():

    try:

        file = request.files.get('file')

        image\_data = request.form.get('imageData')

        if file and allowed\_file(file.filename):

            filename = secure\_filename(file.filename)

            file\_path = os.path.join(app.config['UPLOAD\_FOLDER'], filename)

            file.save(file\_path)

        elif image\_data:

            header, encoded = image\_data.split(",", 1)

            data = base64.b64decode(encoded)

            filename = str(uuid.uuid4()) + ".png"

            file\_path = os.path.join(app.config['UPLOAD\_FOLDER'], filename)

            with open(file\_path, "wb") as f:

                f.write(data)

        else:

            return "No valid image file provided", 400

        analysis\_results = DeepFace.analyze(img\_path=file\_path, actions=['age', 'gender', 'race'], enforce\_detection=False)

        # Check if the result is a list or a dict

        if isinstance(analysis\_results, list):

            # Convert list of dicts to a single dict

            analysis = {key: value for d in analysis\_results for key, value in d.items()}

            analysis\_results = analysis\_results[0]

        else:

            analysis = analysis\_results

        # Extract gender from analysis results

        gender = 'Male' if analysis['gender']['Man'] > analysis['gender']['Woman'] else 'Female'

        return render\_template('results.html', filename=filename, analysis=analysis\_results, gender=gender)

    except Exception as e:

        print("Error occurred: ", traceback.format\_exc())

        return "Error processing the file", 500

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True, threaded=True)

**Home.html:**

<!DOCTYPE html>

<html>

<head>

    <link rel="stylesheet" href="{{ url\_for('static', filename='css/globals.css') }}">

    <link rel="stylesheet" href="{{ url\_for('static', filename='css/home.css') }}">

</head>

<body>

    <div class="cylindrical">

        <div class="div-jsx">

            <header class="header">

                <div class="span-jsx"><div class="w-hoo">WHOO?</div></div>

                <div class="div">

                    <div class="nav">

                        <div class="link"><div class="text-wrapper">Home</div></div>

                        <div class="link-margin"><div class="link-results">Results</div></div>

                    </div>

                    <div class="link-2"></div>

                </div>

            </header>

        </div>

        <div class="div-jsx-wrapper">

            <div class="div-jsx-2">

                <div class="div-jsx-3">

                    <div class="heading">

                        <div class="welcome-let-s">Welcome! Let's<br />analyze the picture</div>

                    </div>

                    <p class="p">Upload your image and get your details</p>

                    <div class="div-jsx-4"></div>

                </div>

                <div class="div-wrapper">

                    <div class="link-wrapper">

                        <div class="let-s-go-wrapper"><div class="let-s-go">Let's go!</div></div>

                    </div>

                </div>

            </div>

        </div>

        <div class="div-jsx-5">

            <form action="{{ url\_for('upload\_file') }}" method="post" enctype="multipart/form-data">

                <input type="hidden" name="imageData" id="imageData">

                <div class="div-jsx-6">

                    <div class="heading-get-your">Get your details</div>

                    <p class="text-wrapper-2">Find out your age, gender, and race</p>

                    <div class="div-jsx-7">

                        <input type="file" id="uploadImage" name="file" accept="image/\*" onchange="previewImage(event)">

                        <label for="uploadImage" class="let-s-go-wrapper"><div class="let-s-go">Upload Image</div></label>

                        <button type="button" id="takePhotoButton" class="let-s-go-wrapper"><div class="let-s-go">Take a Photo</div></button>

                    </div>

                </div>

                <div class="div-jsx-8"></div>

                <div class="link-3">

                    <input type="submit" value="Analyze" class="text-wrapper-4">

                </div>

            </form>

            <img id="imagePreview" style="max-width: 500px; display: none;">

            <video id="camera" width="640" height="480" autoplay style="display:none;"></video>

            <canvas id="canvas" width="640" height="480" style="display:none;"></canvas>

        </div>

        <div class="footer-wrapper">

            <footer class="footer">

                <div class="div-jsx-9">

                    <div class="div-wrapper-2"><div class="text-wrapper-5">WHOO?</div></div>

                    <div class="nav-2">

                        <div class="overlap-group">

                            <div class="div-wrapper-2"><div class="text-wrapper-6">Home</div></div>

                            <div class="div-jsx-10"></div>

                        </div>

                        <div class="span-jsx-2"><div class="text-wrapper-7">Results</div></div>

                    </div>

                </div>

                <div class="div-jsx-11"></div>

                <div class="div-jsx-12"><div class="div-jsx-13"></div></div>

            </footer>

        </div>

    </div>

    <script>

        document.getElementById('uploadImage').addEventListener('change', function(event) {

            var reader = new FileReader();

            reader.onload = function() {

                var output = document.getElementById('imagePreview');

                output.src = reader.result;

                output.style.display = 'block';

                document.getElementById('camera').style.display = 'none';

            };

            reader.readAsDataURL(event.target.files[0]);

        });

        const video = document.getElementById('camera');

        const canvas = document.getElementById('canvas');

        const context = canvas.getContext('2d');

        const takePhotoButton = document.getElementById('takePhotoButton');

        const imageDataInput = document.getElementById('imageData');

        navigator.mediaDevices.getUserMedia({ video: true }).then(function(stream) {

            video.srcObject = stream;

            video.play();

            video.style.display = 'block';

        });

        takePhotoButton.addEventListener('click', function() {

            canvas.width = video.videoWidth;

            canvas.height = video.videoHeight;

            context.drawImage(video, 0, 0);

            var dataURL = canvas.toDataURL('image/png');

            imageDataInput.value = dataURL;

            document.getElementById('imagePreview').src = dataURL;

            document.getElementById('imagePreview').style.display = 'block';

            video.style.display = 'none';

            canvas.style.display = 'none';

            video.srcObject.getTracks().forEach(track => track.stop());

        });

        document.querySelector("form").onsubmit = function() {

            if (imageDataInput.value) {

                document.getElementById('uploadImage').value = '';

            }

        };

    </script>

</body>

</html>

**Results.html:**

<!DOCTYPE html>

<html>

<head>

    <link rel="stylesheet" href="{{ url\_for('static', filename='css/globals.css') }}">

    <link rel="stylesheet" href="{{ url\_for('static', filename='css/results.css') }}">

</head>

<body>

    <div class="cylindrical">

        <div class="div-jsx">

            <header class="header">

                <div class="div-wrapper"><div class="w-hoo">WHOO?</div></div>

                <div class="div">

                    <div class="nav">

                        <div class="link"><div class="text-wrapper">Home</div></div>

                    </div>

                    <div class="link-2"></div>

                </div>

            </header>

        </div>

        <div class="div-jsx-wrapper">

            <div class="div-jsx-2">

                <div class="div-jsx-3">

                    <div class="heading">

                        <p class="here-are-the-results">Here are the results for<br />the uploaded image:</p>

                    </div>

                    <div class="div-jsx-4"></div>

                    <img src="{{ url\_for('static', filename='uploads/' + filename) }}" alt="Uploaded Image" class="image-frame">

                    {% if analysis %}

                    <p class="result-text">Age: {{ analysis['age'] }}</p>

                    <p class="result-text">Gender: {{ gender }}</p>

                    <p class="result-text">Race: {{ analysis['dominant\_race'] }}</p>

                    <p class="result-text">Race Probabilities:</p>

                    <ul class="race-list">

                        {% for race\_name, probability in analysis['race'].items() %}

                            <li>{{ race\_name.title() }}: {{ '%.2f' | format(probability) }}%</li>

                        {% endfor %}

                    </ul>

                {% else %}

                    <p class="no-data">No analysis data available.</p>

                {% endif %}

                    <a href="{{ url\_for('home') }}" class="upload-link">Upload another image</a>

                </div>

            </div>

        </div>

        <div class="footer-wrapper">

            <footer class="footer">

                <div class="div-jsx-6">

                    <div class="span-jsx"><div class="text-wrapper-2">WHOO?</div></div>

                    <div class="nav-2">

                        <div class="div-wrapper"><div class="text-wrapper-3">Home</div></div>

                        <div class="div-jsx-7"></div>

                    </div>

                </div>

                <div class="div-jsx-8"><div class="div-jsx-9"></div></div>

                <div class="div-jsx-10"></div>

            </footer>

        </div>

    </div>

</body>

</html>

## CHAPTER-5

#### RESULTS AND DISCUSSION:

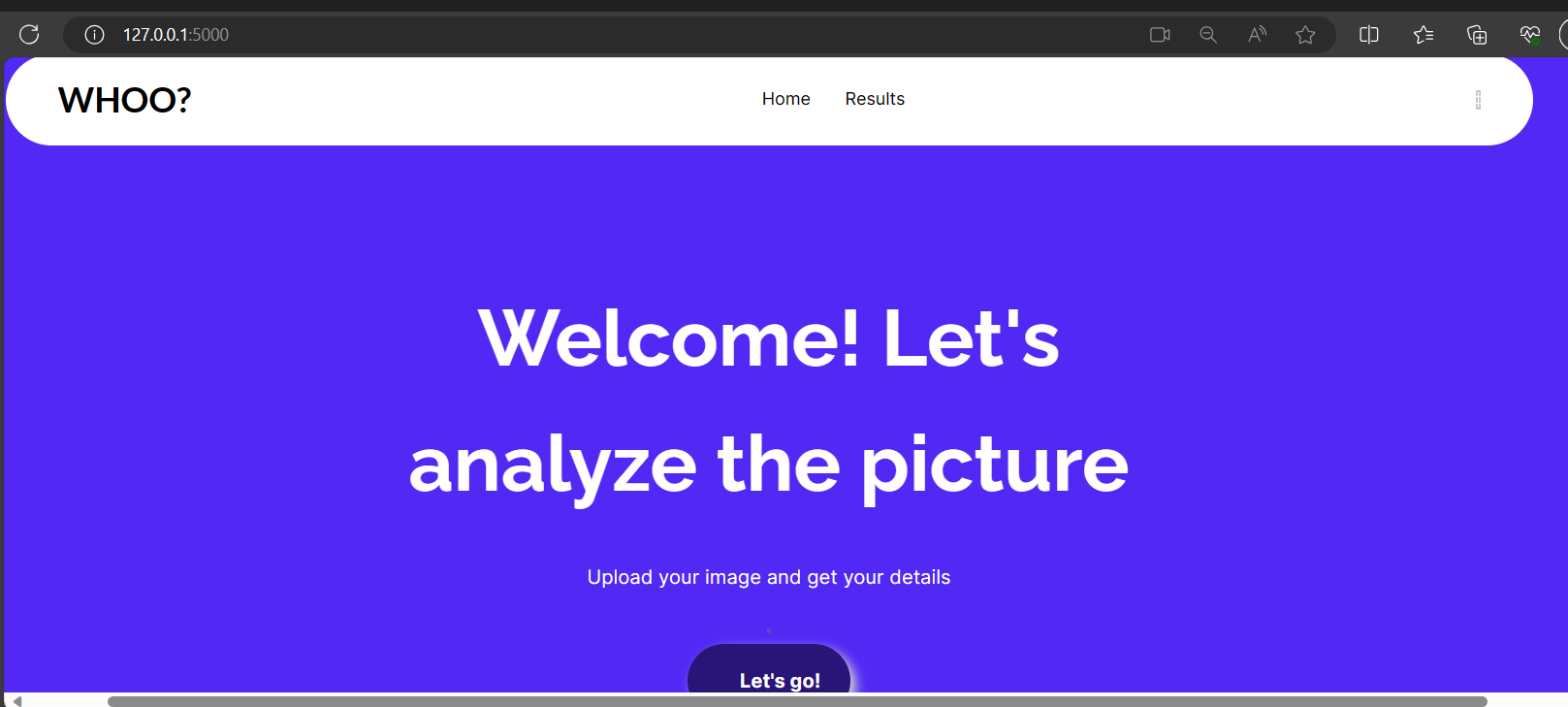


Fig. 4. Home Page of the website

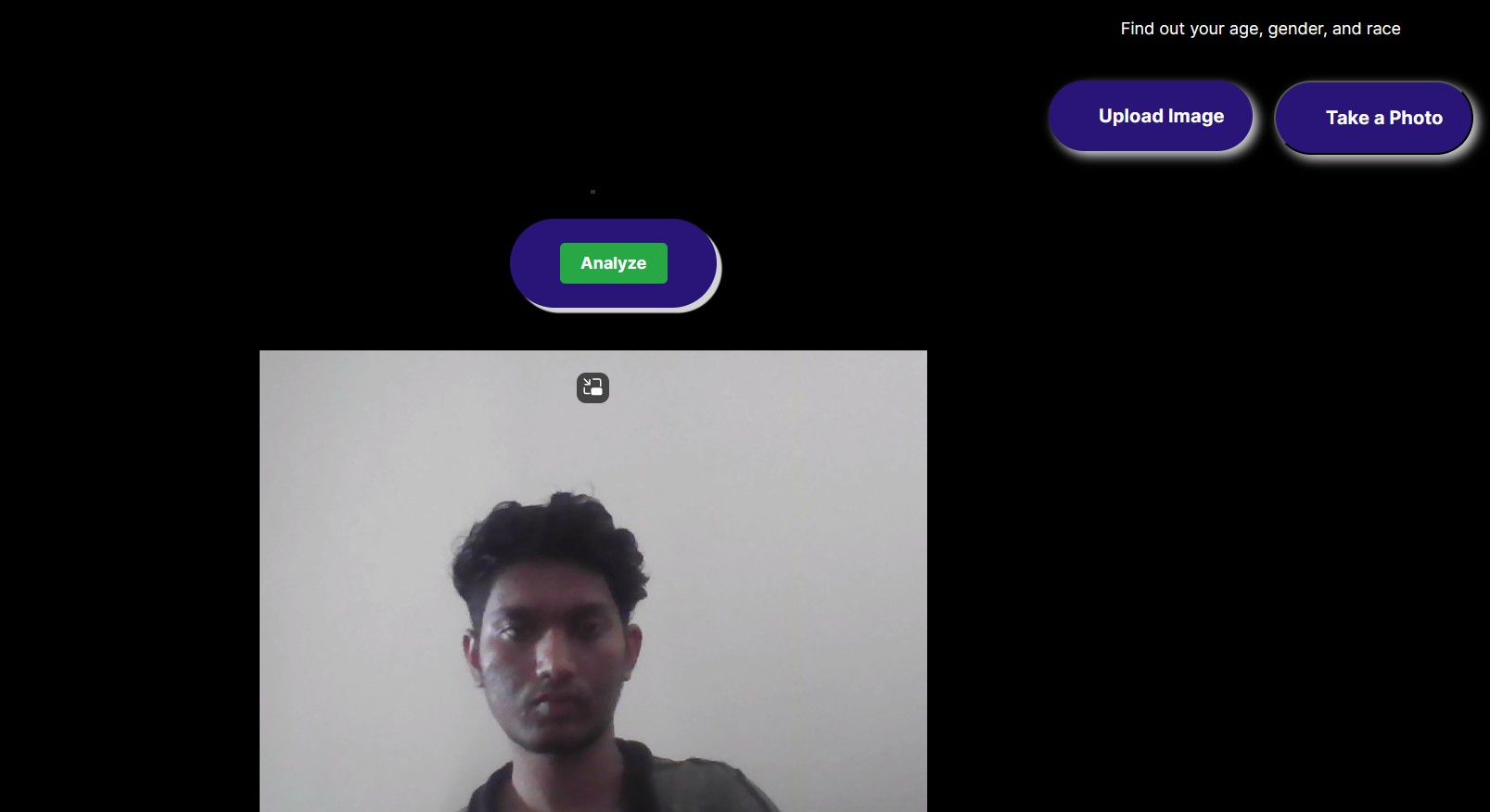


Fig. 5. Home Page of the website

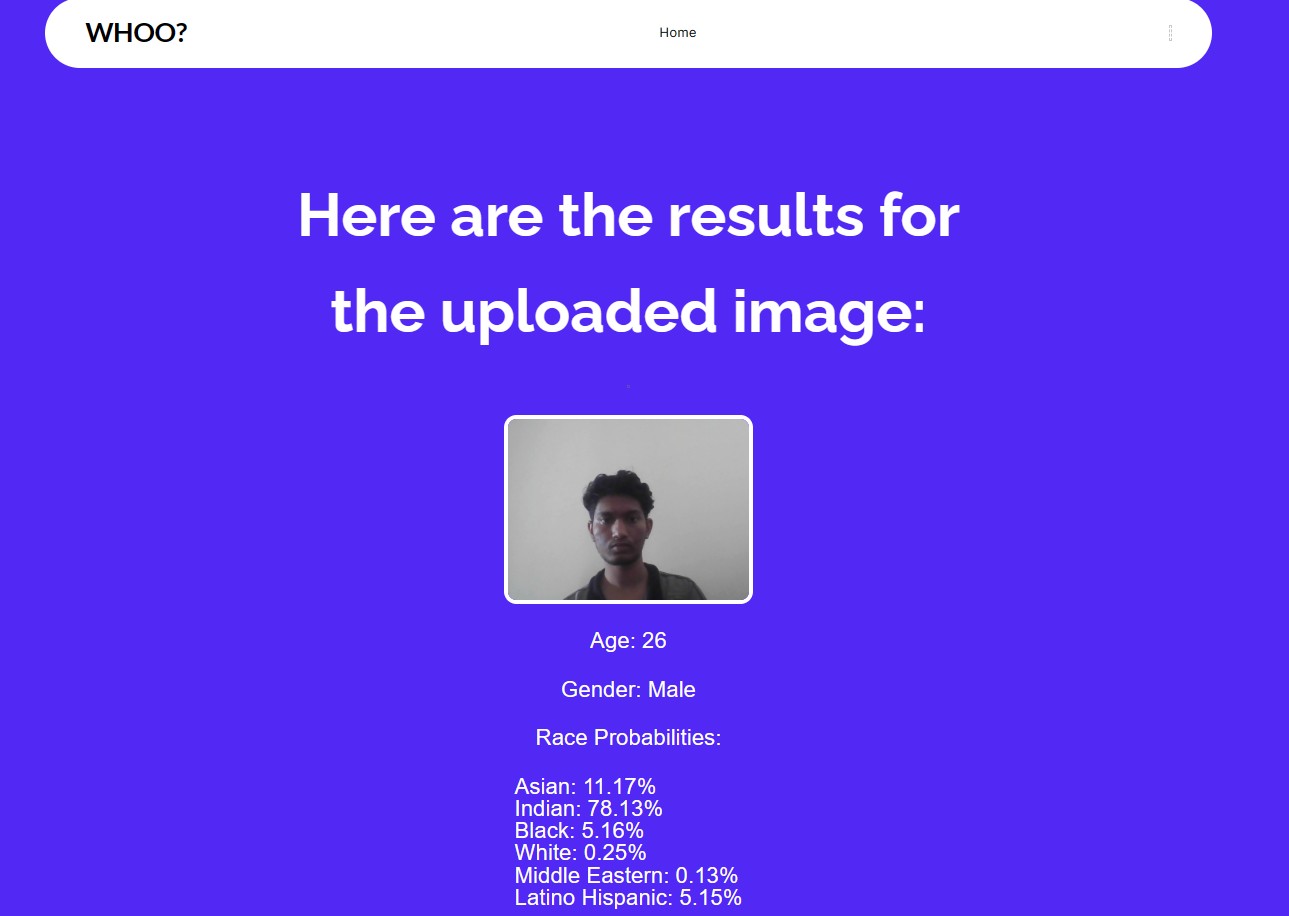


Fig. 6. Results Page

Trying with the different mages with varying lighting conditions and different test faces, we can conclude these criteria about the performance of the model:

DeepFace shows varying accuracy in different age groups, particularly less precise for very young [0-16] and very old [90-116] ages. The model's performance is hindered by foreground objects blocking the face. Accuracy in age and race recognition is significantly affected by lighting and image filters. The model only recognizes gender in binary terms, which may not cover all gender identities. High-quality images are required for accurate recognition, with a minimum dimension requirement of 150 pixels. The presence of glasses is negatively impacting the model's face recognition accuracy. Effective processing requires images to be larger than 150 pixels in width or height. Watermarks in images can disrupt the model's ability to recognize faces. The model's accuracy varies with portrait photos, especially affected by the depth of field.

#### 5.1. Discussion of performance of the models:

#### We have two models, one is a custom-built CNN model and the other is an already existing algorithm called *Deepface*, After testing the models through the datasets, here are the detailed comparison of how two models fared against each other testing the three key attributes: age, gender and race.

#### The dataset that was used for demonstrating the accuracy for both these models is the *UTKFace* by *Susanq.* The present repository contains a wide range of age groups, from 1 to 116, predominant races, and their gender. The following graphs depict the spread of the attributes among the dataset.

#### A pie chart with numbers and text Description automatically generated

Fig. 7. Fig. 8.

\*Others in the *Race Distribution* include: Hispanic, Latino, Middle Eastern.

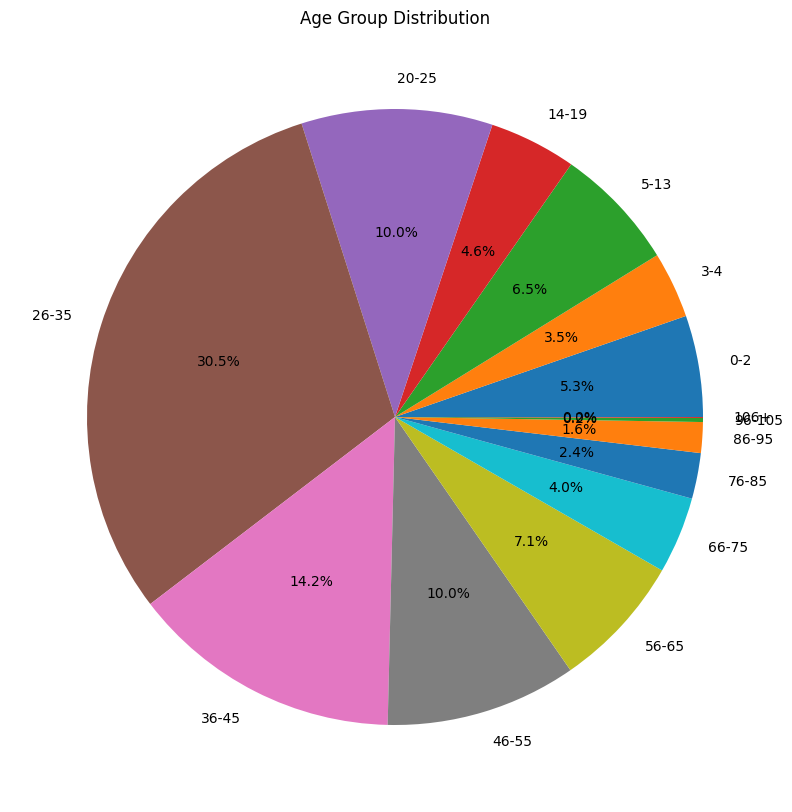


Fig. 9.

Now, let’s analyze the accuracies achieved for each of the attributes by the two models.

**Age:**

**Custom CNN Model:**

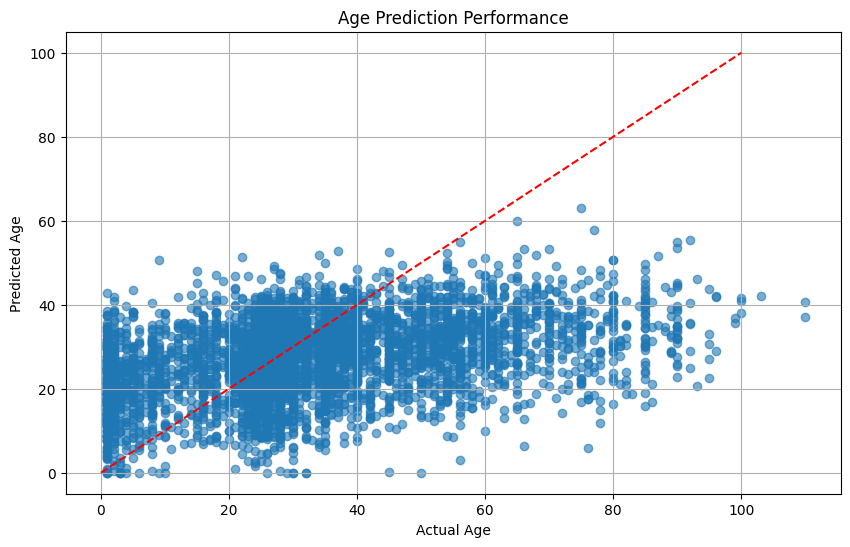


Fig. 10.

Analyzing the graph, we arrive at the following observations:

A dense concentration of data points is visible along the diagonal, indicating many accurate predictions. Disparity between predicted and actual ages increases as the actual age increases, indicated by the vertical spread of data points. The red dashed line represents the perfect prediction accuracy, and we can learn that the model is better at predicting younger ages than older ages. The model appears to underestimate the age for older individuals and overestimate it for younger individuals.

MAE achieved: 14.18902297991213

**Deepface Model:**

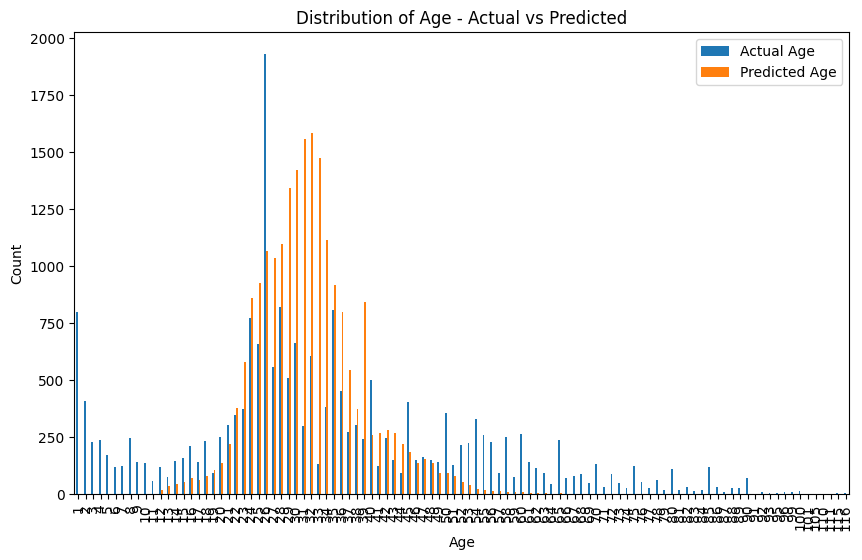
****

Fig. 11

These are the observations that can be drawn from the graph:

There's a prominent peak in the distribution at the 20-30 age range, suggesting a higher concentration of both actual and predicted ages in this group. The actual age distribution appears to be slightly right skewed, while the predicted age distribution closely follows the actual age trend. There is noticeable overestimation in the predicted ages for the 0-10 age range. The distributions diverge as age increases, especially after the 60-age mark, indicating potential inaccuracies in age prediction for older demographics.

MAE achieved: 12.32

After analyzing both the graphs, we can conclude that both models still tend to overestimate the age for younger age groups and underestimate for older age groups, with the deepface model faring slightly better in terms of MAE.

**Gender:**

**Custom Model:**

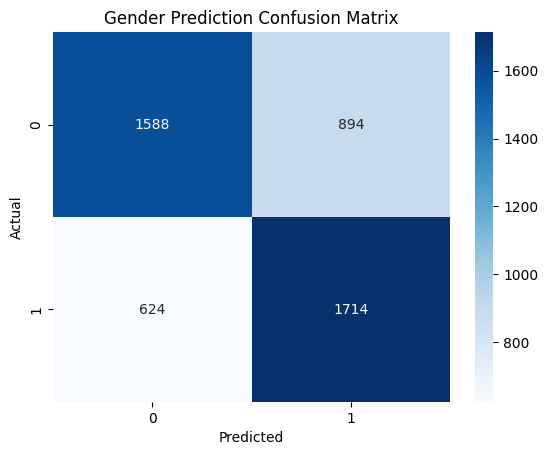


Fig. 12.

Analyzing the graph, we arrive at the following observations:

The confusion matrix indicates a binary classification model for gender prediction, with two classes: 0(male) and 1(female). True Positives: The model correctly predicted 'Male' for 1,588 instances. True Negatives: The model correctly predicted 'Female' for 1,714 instances. False Positives: The model incorrectly predicted 894 instances as 'Female' when they were ‘Male’. False Negatives: The model incorrectly predicted 624 instances as 'Male' when they were 'Female'.

Accuracy for Gender Prediction: 0.6850622406639004

**Deepface Model:**

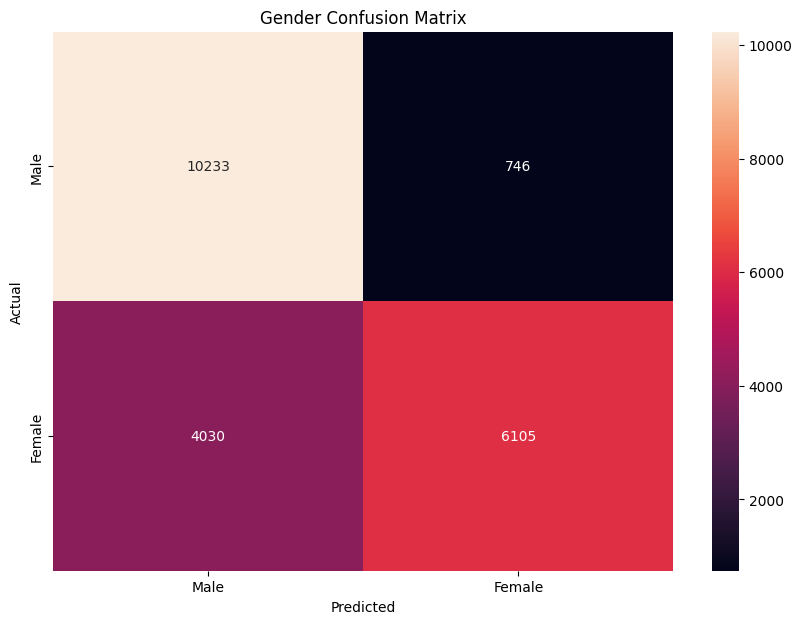


Fig. 13

True Positives: The model correctly predicted 'Male' for 10,233 instances. True Negatives: The model correctly predicted 'Female' for 6,105 instances. False Positives: The model incorrectly predicted 746 instances as 'Female' when they were ‘Male’. False Negatives: The model incorrectly predicted 4,030 instances as 'Male' when they were 'Female'.

Accuracy achieved: 0.773823756

After, analyzing both the confusion matrix, we can conclude that the custom model is better at recognizing Females better than Males, but in contrary, the Deepface model tends to claim better accuracy while classifying Males than Females, with the deepface model faring better in terms of accuracy.

**Race:**

**Custom Model:**

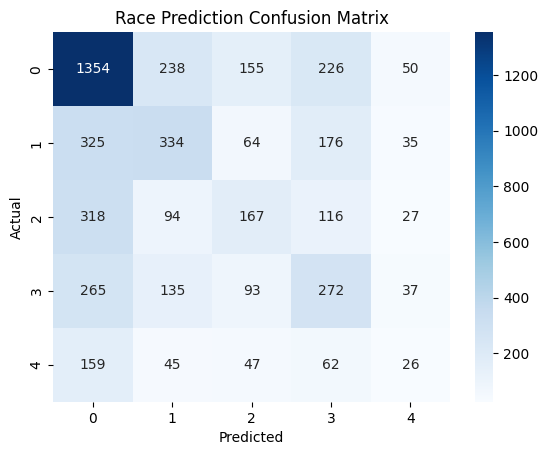


Fig. 14.

Class 0 has the highest number of correct predictions (1354), but also significant misclassifications with other classes, especially with class 3 (226) and class 1 (238). Class 1 shows a relatively balanced distribution of misclassifications with other classes, suggesting less distinctive features learned for this category. Class 2 has lower correct predictions (167) and is often confused with class 0 (318) and class 3 (116). Class 3 has a considerable number of correct predictions (272), with notable confusion with class 0 (265) and class 1 (176). Class 4 shows the least number of correct predictions (62), indicating potential difficulty for the model to distinguish this category or fewer instances in the training data.

Accuracy achieved: 0.4466804979253112

**Deepface Model:**

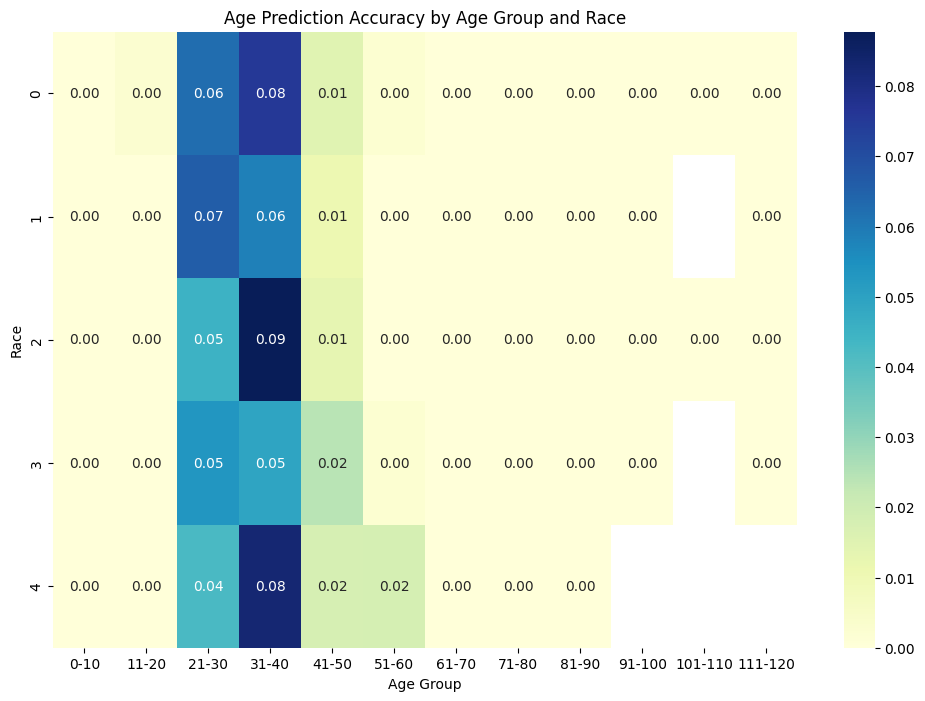


Fig. 15

The age prediction accuracy varies significantly across different racial categories and age groups. The most accurate predictions occur in the age groups of 21-30 and 31-40, across all racial categories presented. Racial category 2 in the age group 31-40 shows the highest accuracy, with a score of 0.09. Racial categories 0 and 4 have lower accuracy scores, particularly in the age groups 41-50 and 51-60. There is a general trend of declining accuracy as age increases, with very little or no accuracy in the oldest age groups (71-120).

Accuracy achieved: 0.61599642681

After analyzing both the graphs, we see that the custom model is better at classifying *Class 1, ‘White’,* solidified with the fact that it is the category with the most instances. Also, the *Class 4, ‘Others’*, *signifying* the need for providing more instances in the dataset to provide a more accurate prediction, with the deepface model faring better in terms of accuracy.

The table here consolidates the results obtained by both models across different attributes.

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithms | Attributes | | |
| Age (MAE) | Gender (Accuracy) | Race (Accuracy) |
| Custom Model | 14.667972297991 | 0.6850622406639004 | 0.4466804979253112 |
| DeepFace | 12.32 | 0.773823756 | 0.615996426 |

## CHAPTER-6

#### CONCLUSION AND FUTURE WORK:

In conclusion, the development and implementation of a Facial Attribute Analysis system using Deepface and CNNs represent a significant stride in the field of artificial intelligence. This system demonstrates the power of deep learning in extracting and interpreting complex facial features, offering a range of applications from enhanced security measures to personalized user experiences. The use of Deepface for Facial Recognition has proven reasonably particularly effective, combining advanced recognition capabilities with robust and accurate attribute analysis.

##### Key Achievements:

The system achieves good accuracy in facial attribute analysis, addressing diverse scenarios and data variations. The intuitive user interface and efficient processing ensure a seamless experience for users of varying technical backgrounds.

##### Future Work:

Future efforts will focus on improving efficiency, and ability to recognize more diverse images with the help of algorithms, reducing computational requirements while maintaining accuracy. Continuously expanding the dataset to include more diverse facial attributes will further improve the system's robustness.Implementing real-time processing capabilities will open new avenues for applications in dynamic environments. Ongoing work will involve navigating the ethical considerations surrounding facial recognition technology.

## CHAPTER-7

#### REFERENCES:

1. "FaceNet: A Unified Embedding for Face Recognition and Clustering" by Florian Schroff, Dmitry Kalenichenko, and James Philbin (2015):
2. "DeepFace: Closing the Gap to Human-Level Performance in Face Verification" by Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf (2014):
3. "SphereFace: Deep Hypersphere Embedding for Face Recognition" by Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song (2017):
4. "ArcFace: Additive Angular Margin Loss for Deep Face Recognition" by Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou (2019):
5. "DeepID3: Face Recognition with Very Deep Neural Networks" by Yi Sun, Xiaogang Wang, and Xiaoou Tang (2015):
6. "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks" by Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao (2016):
7. "Deep Residual Learning for Image Recognition" by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016)
8. H. -B. Kim, N. Choi, H. -J. Kwon and H. Kim, "Surveillance System for Real-Time High-Precision Recognition of Criminal Faces From Wild Videos," in IEEE Access, vol. 11, pp. 56066-56082, 2023, doi: 10.1109/ACCESS.2023.3282451.
9. M. Jha, A. Tiwari, M. Himansh and V. M. Manikandan, "Face Recognition: Recent Advancements and Research Challenges," 2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kharagpur, India, 2022, pp. 1-6, doi: 10.1109/ICCCNT54827.2022.9984308
10. I. Masi, Y. Wu, T. Hassner and P. Natarajan, "Deep Face Recognition: A Survey," 2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), Parana, Brazil, 2018, pp. 471-478, doi: 10.1109/SIBGRAPI.2018.00
11. A. Firmansyah, T. F. Kusumasari and E. N. Alam, "Comparison of Face Recognition Accuracy of ArcFace, Facenet and Facenet512 Models on Deepface Framework," 2023 International Conference on Computer Science, Information Technology and Engineering